# E-POOL, Carsharing and Carpooling Service

# Gruppo "MAI ESS KEW ELL"

Giacomo Minello, Matteo Tramontano, Davide Menetto 26 aprile 2019

#### 1 Introduzione

IL PROGETTO E-POOL, assegnato per il corso di Basi di Dati 2018/2019, consiste nella progettazione ed implementazione di sistema informativo con interfaccia web al fine di gestire i dati di una piattaforma di carsharing e carpoling.

IL BACKEND del progetto si basa sullo stack XAMPP e in aggiunta il DBMS MongoDB per il log delle operazioni di inserimento. Per il frontend ci siamo avvalsi delle tecnologie più comunemente utilizzate (HTML+JS/JQuery).

NONOSTANTE il focus del progetto sia la realizzazione della base di dati abbiamo prestato attenzione anche alla UX, alla UI e all'implementazione di un backend di qualità. In questi ultimi ambiti riteniamo ci siano grandi margini di miglioramento ma considerando che il tempo impiegato per curare questi aspetti ha contribuito a migliorare l'usabilità ed abbia facilitato la verifica della correttezza dell'implementazione della base di dati riteniamo a posteriori che la scelta sia stata ottimale.

#### 2 Analisi delle specifiche

La traccia completa del progetto è disponibile a questo indirizzo.

#### 2.1 OPERAZIONI

Nel progetto vengono gestite le seguenti operazioni per tutti i tipi di utente:

- Registrazione di un utente
- Registrazione di un utente aziendale
- Autenticazione di un utente
- Autenticazione di un utente aziendale
- Visualizzare i dati di altri utenti
- Effettuare una prenotazione di un veicolo
- Visualizzare lo storico delle prenotazioni effettuate
- Visualizzare i veicoli disponibili
- Visualizzare i dati delle società di car-sharing
- Visualizzare i dati delle aziende
- Inserire una segnalazione sullo stato di un veicolo

Gli utenti premium o dipendenti di un'azienda possono eseguire anche le seguenti operazioni:

- Inserire un tragitto e condividerlo
- Visualizzare i tragitti prenotabili
- Aggiungersi ad una prenotazione esistente
- Inserire una valutazione di un utente con cui si è condiviso un viaggio

A tutti gli utenti sarà possibile accedere alle seguenti operazioni sui dati:

- Visualizzare la classifica degli utenti sulla base del voto medio ricevuto
- Visualizzare la classifica degli utenti più attivi in base al numero di segnalazioni
- Visualizzare la classifica dei modelli di veicolo più prenotati

2.2 VOLUMI

<i>Entit</i> à	Volume	
Azienda	10	
Foto	1000	
Passaggio	100	
Pdf	200	
Prenotazione	1000	
Privata	10	
Pubblica	10	
Segnalazione	1000	
Società	10	
Sosta	100	
Тарра	1000	
Tragitto	100	
Tragitto Prenotazione	10000	
Utente	500	
Utente Aziendale	100	
<b>Utente Premium</b>	100	
<b>Utente Semplice</b>	300	
Valutazione	500	
Veicolo	200	
	<u> </u>	

# 3 PROGETTAZIONE

# 3.1 PROGETTAZIONE CONCETTUALE

## 3.1.1 DIAGRAMMA E-R

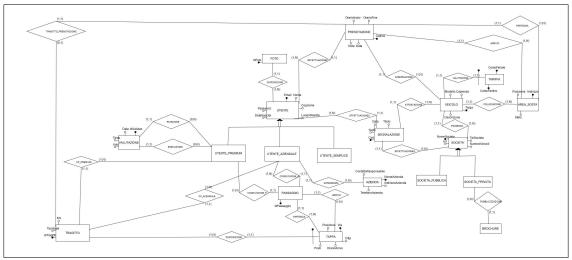


Figura 3.1: Diagramma E-R generato con JDER

Il file originale del diagramma e una copia in formato immagine sono disponibili negli allegati.

### 3.1.2 DIZIONARIO ENTITÀ E RELAZIONI

<i>Entit</i> à	Attributi	Identificatore
azienda	NOME, INDIRIZZO, TELEFONO, RECAPITO	NOME
foto	IDFOTO, EMAIL-UTENTE, PATHFOTO	IDFOTO
passaggio	ID-PASSAGGIO, ID-TAPPA, EMAILP, EMAI-	ID-PASSAGGIO
	LA, INDIRIZZO-PARTENZA, INDIRIZZO- ARRIVO	
pdf	IDPDF, NOME-SOCIETA, PATH	IDPDF
prenotazione	ID, INIZIO, FINE, NOTE, AUTO,	ID
•	UTENTE, INDIRIZZO-PARTENZA,	
	INDIZIZZO-ARRIVO	
privata	NOME	NOME
pubblica	NOME	NOME
segnalazione	ID, EMAIL, SOCIETA, DATA, TITOLO, TESTO,	ID
	AUTO	
societa	NOME, URL, TELEFONO	NOME
sosta	INDIRIZZO, LAT, LNG, RICARICA	INDIRIZZO
tappa	ID-TRAGITTO, CITTA, VIA, LAT	ID-TRAGITTO, VIA
tragitto	ID, EMAILP, EMAILA, KM, TIPO	ID
tragitto-prenotazione	ID-TRAG, ID-PREN	ID-TRAG, ID-PREN
utente	EMAIL, PW, NOME, COGNOME, DATANA-	<b>EMAIL</b>
	SCITA, LUOGO	
utente-aziendale	EMAILA, NOMEAZIENDA	<b>EMAILA</b>
utente-premium	EMAILP	EMAILP
utente-semplice	EMAILS	<b>EMAILS</b>
valutazione	ID, EMAIL, DATA, TESTO, VOTO, UTENTE	ID
veicolo	TARGA, MODELLO, CAPIENZA, DESCRIZIO-	TARGA
	NE, FERIALE, FESTIVO, SOCIETA, AREA-	
	SOSTA, STATO	

Relazione	Descrizione	Molteplicità	Componenti
Disposizione	Associa n foto ad un utente	(1N)-(11)	UTENTE-FOTO
EffettuazioneU	Associa n segnalazioni ad un utente	(1N-11)	UTENTE-SEGNALAZIONE
EffettuazioneP	Associa n prenotazioni ad un utente	(1N-11)	UTENTE-PRENOTAZIONE
Assegnazione	Associa n prenotazioni ad un veicolo	(11-1N)	PRENOTAZIONE-VEICOLO
Attestazione	Associa n segnalazioni ad un veicolo	(11-1N)	SEGNALAZIONE-VEICOLO
Partenza	Associa n prenotazioni ad un area di sosta	(11-1N)	PRENOTAZIONE-AREA SOSTA
Arrivo	Associa n prenotazioni ad un area di sosta	(11-1N)	PRENOTAZIONE-AREA SOSTA
Valutazione	Associa un veicolo ad una tariffa	(11-11)	VEICOLO-TARIFFA
Collocazione	Associa n veicoli ad un area di sosta	(11-1N)	VEICOLO-AREA SOSTA
Possesso	Associa n veicoli ad una società	(11-1N)	VEICOLO-SOCIETA'
EffettuazioneS	Associa n segnalazioni ad una società	(11-1N)	SEGNALAZIONE-SOCIETA'
Pubblicizzazione	Associa n brochure ad una società privata	(1N-11)	SOCIETA' PRIVATA-BROCHUR
CondivisioneA	Associa n passaggi ad un utente aziendale	(1N-11)	UTENTE AZIENDALE-PASSAGGIO
Dipendenza	Associa n utenti aziendali ad una azienda	(11-1N)	UTENTE AZIENDALE-AZIENDA
Partenza	Associa n tappe ad un passaggio	(11-1N)	PASSAGGIO-TAPPA
Arrivo	Associa n tappe ad un passaggio	(11-1N)	PASSAGGIO-TAPPA
CondivisioneP	Associa n passaggi ad un utente premium	(1N-11)	UTENTE PREMIUM-PASSAGGIO
Ricezione	Associa n valutazioni ad un utente premium	(0N-11)	UTENTE PREMIUM-VALUTAZIONE
Esecuzione	Associa n valutazioni ad un utente premium	(0N-11)	UTENTE PREMIUM-VALUTAZIONE
CpAziendale	Associa n tragitti ad un utente aziendale	(1N-11)	UTENTE AZIENDALE-TRAGITTO
CpPremium	Associa n tragitti ad un utente premium	(1N-11)	UTENTE PREMIUM-TRAGITTO
Disposizione	Associa n tappe ad un tragitto	(11-1N)	TAPPA-TRAGITTO
TragittoPrenotazione	Associa uno o zero tragitti ad una prenotazione	(11-01)	PRENOTAZIONE-TRAGITTO

#### 3.1.3 Business rules

Vengono considerate le seguenti business rules:

- 1. La prenotazione è possibile solo su veicoli non già prenotati da altri utenti per la stessa data/orario
- 2. Tenere traccia dell'ordine delle tappe nel tragitto
- 3. Condivisione spese per una specifica prenotazione
- 4. Richiesta di condivisione accettata automaticamente a patto che non si ecceda la capacità del veicolo
- 5. Ogni tragitto può essere di tipo: URBANO, EXTRA-URBANO, AUTOSTRADALE, MISTO
- 6. Ogni area di sosta può essere dotata o meno di una colonnina di ricarica

#### 3.2 PROGETTAZIONE LOGICA

#### 3.2.1 RISTRUTTURAZIONE SCHEMA CONCETTUALE

Tutte le relazioni uno-a-molti sono state trasformate in attributi. Relazioni di altro tipo sono state tradotte come tabelle. Le generalizzazioni sono state tradotte con lo scopo di minimizzare i valori null a discapito del numero di tabelle.

#### 3.2.2 RIDONDANZE

Assumendo i seguenti volumi:

- Numero medio società -> 10
- Numero medio veicoli ->100
- Numero medio veicoli per società -> 20

Assumendo il seguente peso delle operazioni:

- Peso operazioni interattive -> 0.8 ( $W_i$ )
- Peso operazioni batch -> 0.2 ( $W_b$ )
- Peso operazioni scrittura -> 2 ( $\alpha$ )

Assumendo le seguenti operazioni sui dati:

- Inserire nuovo veicolo per una specifica società -> 1 volta al mese, interattiva (OP 1)
- Rimuovere un veicolo per una specifica società -> 1 volta al mese, interattiva (OP 2)
- Contare i veicoli per una specifica società -> 5 volte al mese, batch (OP 3)

Senza ridondanza si ha che:

ор	frequenza	peso	letture	scritture	costo
1	1	0.8	0	2	3.2
2	1	8.0	0	2	3.2
3	5	0.2	40	0	40

con un costo totale pari a

$$C(S) = 46.4 \tag{3.1}$$

Considerando la ridondanza invece si ha che:

ор	frequenza	peso	letture	scritture	costo
1	1	0.8	0	2	3.2
2	1	8.0	0	2	3.2
3	5	0.2	1	0	1

con un costo totale pari a

$$C(Srid) = 7.2 \tag{3.2}$$

Procedento a calcolare lo speedup si ha che

$$\frac{C(S)}{C(Srid)} = \frac{46.4}{7.2} = 6.44\tag{3.3}$$

Questo valore di SpeedUp indica che è lievemente conveniente introdurre il campo numero veicoli nella tabella SOCIETA'

La documentazione completa del database è disponibile in formato HTML negli allegati alla cartella "epool@localhost".

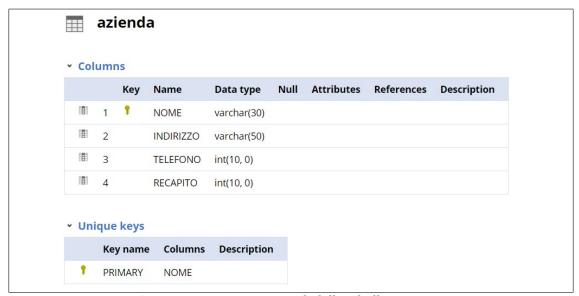


Figura 3.2: Struttura e vincoli della tabella AZIENDA

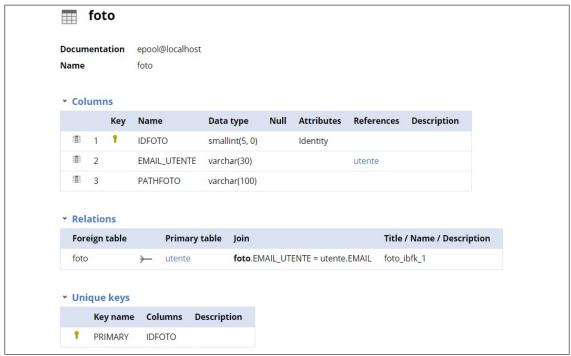


Figura 3.3: Struttura e vincoli della tabella FOTO

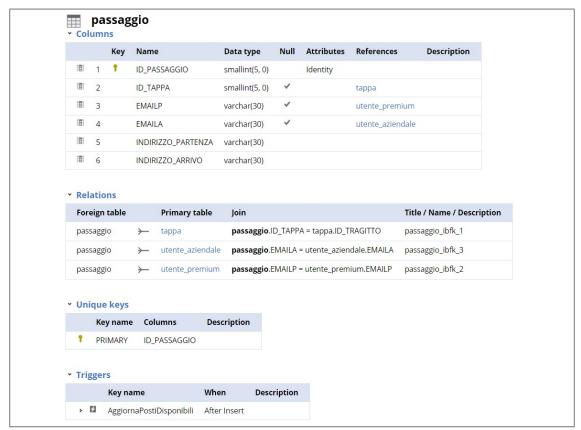


Figura 3.4: Struttura e vincoli della tabella PASSAGGIO

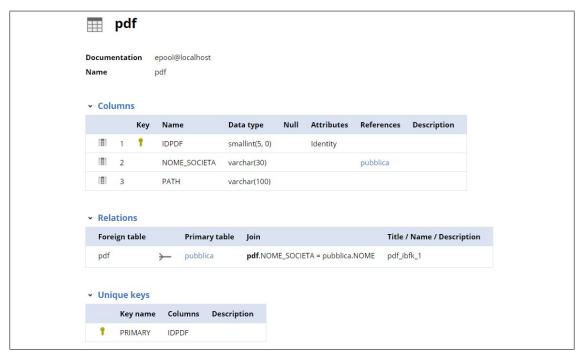


Figura 3.5: Struttura e vincoli della tabella PDF

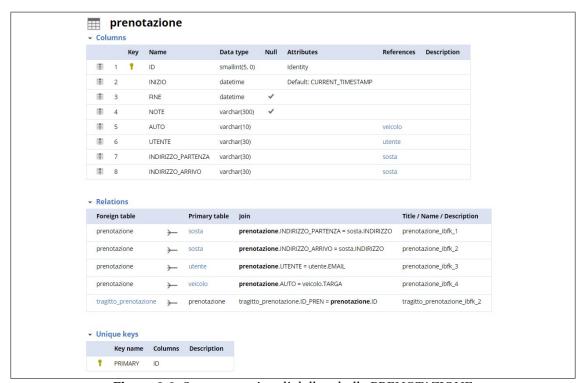


Figura 3.6: Struttura e vincoli della tabella PRENOTAZIONE

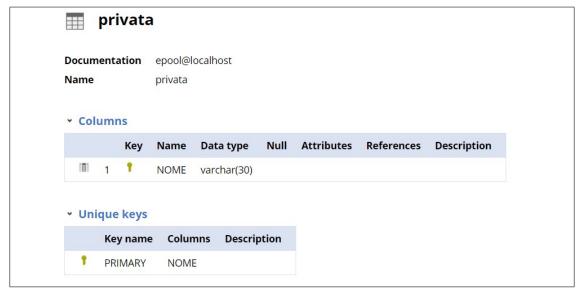


Figura 3.7: Struttura e vincoli della tabella PRIVATA



Figura 3.8: Struttura e vincoli della tabella PUBBLICA

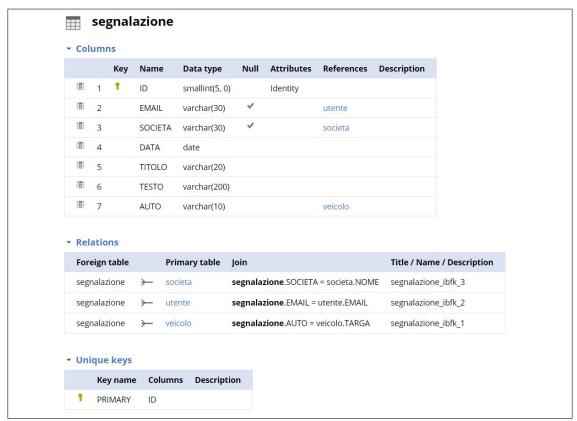


Figura 3.9: Struttura e vincoli della tabella SEGNALAZIONE

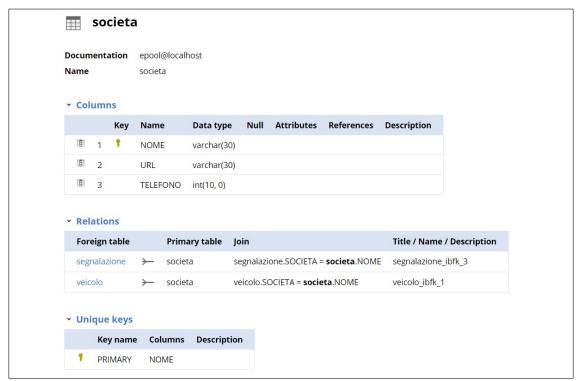


Figura 3.10: Struttura e vincoli della tabella SOCIETA

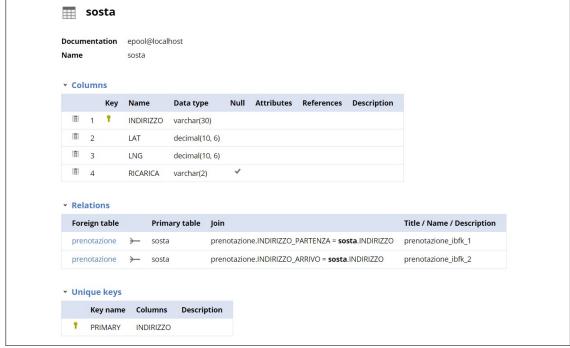


Figura 3.11: Struttura e vincoli della tabella SOSTA

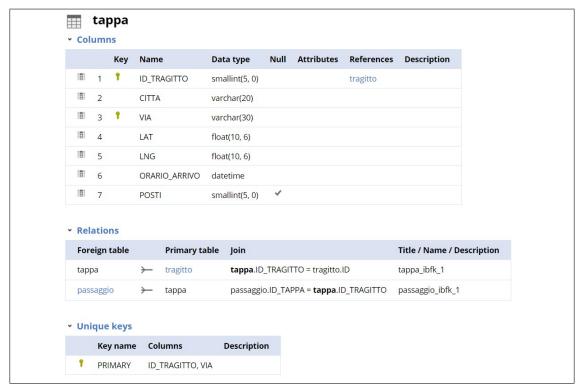


Figura 3.12: Struttura e vincoli della tabella TAPPA

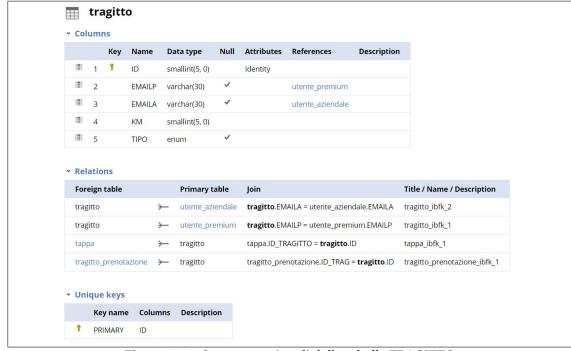


Figura 3.13: Struttura e vincoli della tabella TRAGITTO

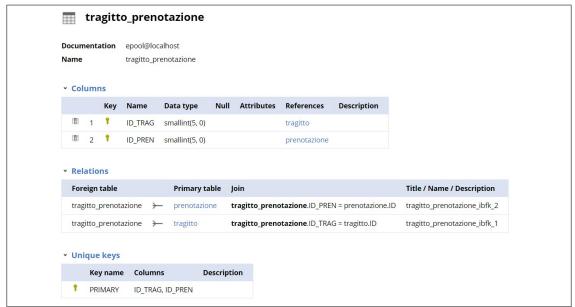


Figura 3.14: Struttura e vincoli della tabella TRAGITTOPRENOTAZIONE

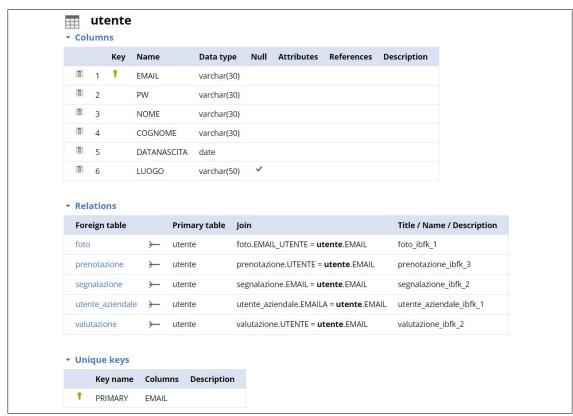


Figura 3.15: Struttura e vincoli della tabella UTENTE



Figura 3.16: Struttura e vincoli della tabella UTENTEAZIENDALE

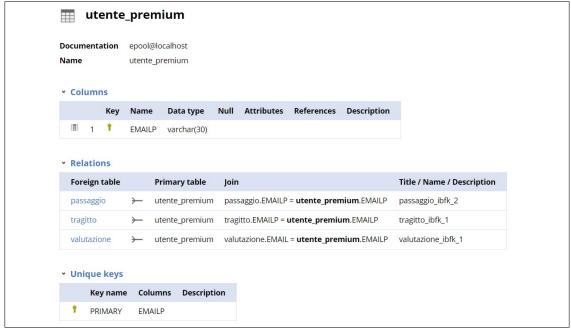


Figura 3.17: Struttura e vincoli della tabella UTENTEPREMIUM



Figura 3.18: Struttura e vincoli della tabella UTENTESEMPLICE



Figura 3.19: Struttura e vincoli della tabella VEICOLO

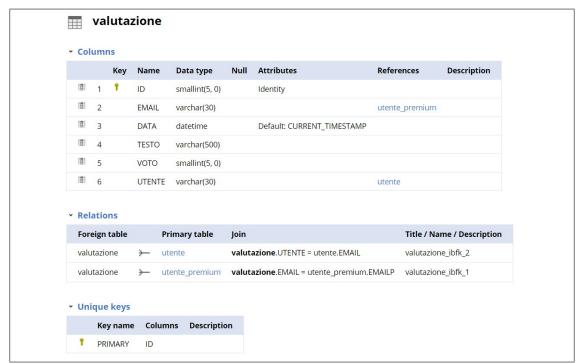


Figura 3.20: Struttura e vincoli della tabella VALUTAZIONE

#### 4 Funzionalità

All'accesso al sito, durante il caricamento della pagina principale, viene mostrata un'ipnotizzante animazione di caricamento. Una volta caricata la pagina principale del sito l'utente ha la possibilità di inserire rapidamente le proprie credenziali per autenticarsi attraverso due form:

- il primo form è dedicato agli utenti semplici e premium
- il secondo form è dedicato agli utenti dipendenti di un'azienda

La scelta di non unificare le due modalità di autenticazione è stata fatta per dare la possibilità di separare gli accessi, immaginando una pagina di accesso dedicata per i dipendenti di un'azienda. "Under the hood" all'accesso al sito viene caricata la pagina *index* che include anche i form per il login. Ciò permette di dare spazio ad ulteriori modifice future e permettere facilmente di riarrangiare la UX del sito, ad esempio inserendo un form per il login ed uno per la registrazione, messaggi di benvenuto, istruzioni per l'utente, ecc..

Qualora l'utente non fosse registrato è possibile accedere ai form di registrazione tramite il menu. Come per il login abbiamo scelto di separare i due metodi di registrazione, sempre tenendo conto di voler mantenere una pagina dedicata per i dipendenti di un'azienda. Qualora l'utente non abbia effettuato il login e provi ad accedere a delle pagine riservate ad utenti già autenticati egli verrà reindirizzato alla pagina principale del sito dove avrà la possibilità di autenticarsi. Allo stesso modo, qualora l'utente abbia effettuato il login e provi ad accedere a delle pagine per le quali non è autorizzato egli verrà reindirizzato alla home (questo caso

si applica in particolare agli utenti semplici che provano ad accedere a funzioni riservate ad utenti premium e ai dipendenti di un'azienda).

Una volta effettuato il login con successo l'utente viene reindizizzato alla home dedicata in base al tipo di utente. Gli utenti semplici e premium condividono la stessa home (avendo però permessi diversi) e i dipendenti di un'azienda hanno una home dedicata. Entrambe le home includono un header, un footer ed un body vuoto (dove si può immaginare vengano mostrate comunicazioni, informazioni, ecc..

Tramite il menu è possibile visualizzare tutte le pagine del sito nella quali i nomi indicano le features disponibili. Le pagine sono raggruppate in:

- home, un link utile per tornare alla pagina principale del sito (sebbene la funzione sia disponibile cliccando il logo nello stesso menu, per ottimizzare l'UX sia in modalità desktop che mobile il logo viene nascosto quando si scorre una pagina).
- utente standard, contiene le funzioni disponibili per tutti gli utenti autenticati.
- premium features, contiene le funzioni disponibili per gli utenti premium e i dipendenti di un'azienda.
- registrazione, permette di accedere ai form di registrazione.
- statistiche, una pagina pubblica che permette di visualizzare alcuni dati relativi alla piattaforma.

Senza entrare nel dettaglio di ogni pagina alcune funzionalità non strettamente relative al database che sono state incluse sono:

- nella pagina del *profilo* utente è possibile caricare delle foto e visualizzarle su uno slideshow (una piccola nota, dopo il caricamento di una foto l'immagine potrebbe non essere immediatamente visibile, per ovviare al problema è possibile disabilitare la cache del sito in modo da forzarne l'aggiornamento).
- nella pagina *veicolidisponibili* è possibile visualizzare i veicoli e la loro posizione su una mappa tramite le API Google (di recente sono cambiate le modalità per ottenere una Key valida per l'utilizzo delle API, imponendo l'obbligo di abilitare la fatturazione di Google Cloud Platform: eventuali problemi di caricamento sono dovuti all'utilizzo di una chiave non valida che abbiamo utilizzato per limitarci a mostrare la possibilità di implementare questa funzione).
- nella pagina *vi sualizzasociet* à è possibile visualizzare i dati delle società di car sharing ed anche delle brochure pdf se presenti (la possibilità di inserire tali brochure non è stata gestita tramite il sito immaginando che tale funzionalità venga riservata ad un amministratore o alla società stessa; la gestione di questo caso d'uso è quindi presente come stored procedure).

# 5 CODICE MYSQL

Per agevolare la leggibilità del codice SQL ne riportiamo solo alcuni snippets. Il codice completo è disponibile negli allegati.

#### 5.1 VISTE

#### 5.1.1 MEDIA VOTO UTENTE

- 1 CREATE VIEW MEDIA\_VOTO\_UTENTE (UTENTE, MEDIA\_VOTO) AS
- SELECT UTENTE, AVG (VOTO)
- 3 FROM VALUTAZIONE
- 4 GROUP BY UTENTE
- ORDER BY AVG(VOTO) DESC;

#### 5.1.2 PRENOTAZIONE CAPIENZA

- 1 CREATE VIEW PRENOTAZIONE\_CAPIENZA (AUTO, POSTI\_DISPONIBILI, IDTRAGITTO ,GUIDATORE,
- TAPPA\_CITTA, TAPPA\_VIA) AS
- 3 SELECT TARGA, CAPIENZA-1, TRAGITTO.ID, UTENTE, TAPPA.CITTA, TAPPA.VIA
- 4 FROM PRENOTAZIONE, VEICOLO, TRAGITTO\_PRENOTAZIONE, TRAGITTO, TAPPA
- 5 WHERE (AUTO=TARGA AND PRENOTAZIONE.ID = ID\_PREN AND TRAGITTO.ID = ID\_TRAG AND
- 6 ID\_TRAGITTO= TRAGITTO.ID);

#### 5.1.3 VEICOLI DISPONIBILI

- 1 CREATE VIEW VEICOLI\_DISPONIBILI(TARGA, MODELLO, CAPIENZA, DESCRIZIONE, FERIALE,
- FESTIVO, SOCIETA, AREA\_SOSTA) AS
- 3 SELECT TARGA, MODELLO, CAPIENZA, DESCRIZIONE, FERIALE, FESTIVO, SOCIETA, AREA\_SOSTA
- 4 FROM VEICOLO
- WHERE STATO = 'NON IN USO'

#### 5.2 Triggers

#### 5.2.1 AGGIORNA POSTI DISPONIBILI

```
DELIMITER |

CREATE TRIGGER AggiornaPostiDisponibili

AFTER INSERT ON PASSAGGIO

FOR EACH ROW

BEGIN

UPDATE TAPPA

SET TAPPA.POSTI=TAPPA.POSTI-1

WHERE TAPPA.VIA=new.INDIRIZZO_ARRIVO and tappa.ID_TRAGITTO=new.ID_TAPPA;

END

| DELIMITER;
```

#### 5.2.2 BLOCCA VEICOLO

```
DELIMITER |

CREATE TRIGGER BloccaVeicolo

AFTER INSERT ON PRENOTAZIONE

FOR EACH ROW

BEGIN

UPDATE VEICOLO SET STATO = 'IN USO' WHERE TARGA=NEW.AUTO;

END;

DELIMITER;
```

#### 5.2.3 AGGIORNA LIVELLO UTENTE

```
DELIMITER |

CREATE TRIGGER AggiornaLivelloUtente

AFTER INSERT ON PRENOTAZIONE

FOR EACH ROW

BEGIN

IF ((SELECT COUNT(*) FROM prenotazione WHERE UTENTE = NEW.UTENTE) = 3)

THEN

DELETE FROM utente_semplice WHERE EMAILS IN

(SELECT UTENTE FROM PRENOTAZIONE WHERE PRENOTAZIONE.UTENTE = NEW.UTENTE);

INSERT INTO utente_premium (EMAILP)

(SELECT distinct (UTENTE) FROM PRENOTAZIONE WHERE PRENOTAZIONE.UTENTE = NEW.UTENTE);

END IF;

END;
```

#### 5.3 STORED PRECEDURES

#### 5.3.1 LOGIN

```
DELIMITER |

create PROCEDURE Login (IN Email varchar(30), IN pasw varchar(30), OUT result BOOLEAN)

BEGIN

IF EXISTS(SELECT * FROM UTENTE WHERE UTENTE.EMAIL = Email AND UTENTE.PW=pasw) THEN

SET result = (TRUE);

ELSE

SET result = (FALSE);

END IF;

END;

DELIMITER;
```

## 5.3.2 LOGIN AZIENDALE

```
DELIMITER |
1
    create PROCEDURE LoginAziendale (IN Email varchar(30), IN pasw varchar(30),
    OUT result BOOLEAN)
    BEGIN
    IF EXISTS(SELECT * FROM UTENTE WHERE UTENTE.EMAIL = Email AND UTENTE.PW=pasw) THEN
    IF EXISTS(SELECT * FROM UTENTE_AZIENDALE WHERE UTENTE_AZIENDALE.EMAILa = Email) THEN
    SET result = (TRUE);
    SET result = (FALSE);
    END IF;
10
    ELSE
11
    SET result = (FALSE);
12
13
    END IF;
    END;
14
15
    DELIMITER;
16
```

#### 5.3.3 LOGIN TYPE

```
DELIMITER |
create PROCEDURE LoginType (IN Email varchar(30), OUT result BOOLEAN)

BEGIN

IF EXISTS(SELECT * FROM UTENTE_PREMIUM WHERE UTENTE_PREMIUM.EMAILP = Email) THEN

SET result = 2;

ELSE

SET result = 1;

END IF;

END;

DELIMITER;
```

### 5.3.4 REGISTRAZIONE UTENTE

```
DELIMITER |
1
    CREATE PROCEDURE RegistrazioneUtente (IN EmailN varchar(30), IN pasw varchar(30),
2
    IN nome varchar(30), IN cognome varchar (30),
    IN datanascita date, IN luogo varchar (50), OUT result BOOLEAN)
    BEGIN
    start transaction;
    SET result = (FALSE);
    IF NOT EXISTS ( SELECT EMAIL FROM UTENTEWHERE EMAIL = EmailN)
    INSERT INTO UTENTE (EMAIL, PW, NOME, COGNOME, DATANASCITA, LUOGO)
    VALUES (EmailN, pasw, nome, cognome, datanascita, luogo);
11
    SET result = (TRUE);
12
13
    commit work;
    ELSE
14
    CALL printf ('[ERRORE] UTENTE GIA REGISTRATO');
15
    rollback;
16
    END IF;
17
    END;
18
19
    DELIMITER;
```

#### 5.3.5 REGISTRAZIONE DIPENDENTE

```
DELIMITER |
1
    CREATE PROCEDURE RegistrazioneAziendale (IN Emailt varchar(30), IN pasw varchar(30),
2
    IN nome varchar(30), IN cognome varchar (30),
    IN datanascita date, IN luogo varchar (50), IN nomeazienda varchar (30),
    OUT result BOOLEAN)
    BEGIN
    start transaction;
    SET result = (FALSE);
    IF NOT EXISTS ( SELECT EMAIL FROM UTENTE WHERE EMAIL = Emailt)
    THEN IF EXISTS (SELECT * FROM AZIENDA WHERE NOMEAZIENDA = nomeazienda)
11
    INSERT INTO UTENTE (EMAIL, PW, NOME, COGNOME, DATANASCITA, LUOGO)
12
    VALUES (Emailt, pasw, nome, cognome, datanascita, luogo);
13
    INSERT INTO UTENTE_AZIENDALE (EMAILA, NOMEAZIENDA) VALUES (Emailt, nomeazienda);
14
    SET result = (TRUE);
15
    commit work;
16
    ELSE
17
    CALL printf ('[ERRORE] AZIENDA NON PRESENTE NEL SISTEMA');
18
19
    END IF;
    ELSE
21
    CALL printf ('[ERRORE] UTENTE GIA REGISTRATO');
22
    rollback;
23
    END IF;
24
    END;
25
    DELIMITER;
27
```

#### 5.3.6 INSERISCI PRENOTAZIONE PREMIUM/AZIENDALE

```
DELIMITER |
1
   CREATE PROCEDURE InserisciPrenotazioneAziendale (IN Tragitto smallint,
2
    IN Notet VARCHAR(300), IN Automobile VARCHAR(10), IN Emailt VARCHAR(30),
   IN IndirizzoPartenzat VARCHAR(30), IN IndirizzoArrivot VARCHAR(30), OUT result BOOLEAN)
    BEGIN
    start transaction;
6
    SET result = (FALSE);
   INSERT INTO `prenotazione` ( `NOTE`, `AUTO`, `UTENTE`, `INDIRIZZO_PARTENZA`,
    `INDIRIZZO_ARRIVO`)
    VALUES ( Notet, Automobile, Emailt, IndirizzoPartenzat, IndirizzoArrivot);
10
    SET result = (SELECT LAST_INSERT_ID());
11
    INSERT INTO `tragitto_prenotazione` ( `ID_TRAG`, `ID_PREN`) VALUES ( Tragitto, result);
12
13
    SET result = (TRUE);
    commit work;
14
    END;
15
16
    DELIMITER;
17
```

## 5.3.7 TERMINA PRENOTAZIONE

```
DELIMITER |
1
    CREATE PROCEDURE `TerminaPrenotazione`(IN `Email` VARCHAR(30), IN `Auto` VARCHAR(10))
2
3
   DECLARE TEMPO DATETIME DEFAULT NOW();
    start transaction;
    IF EXISTS (SELECT * FROM PRENOTAZIONE WHERE Auto = AUTO AND UTENTE = Email)
6
    THEN
    UPDATE VEICOLO SET STATO = 'NON IN USO' WHERE TARGA = Auto;
    UPDATE VEICOLO SET AREA_SOSTA =
    (SELECT INDIRIZZO_ARRIVO FROM PRENOTAZIONE
10
    WHERE Prenotazione.AUTO = Auto AND FINE is null) WHERE TARGA = Auto;
11
    UPDATE PRENOTAZIONE SET FINE = TEMPO WHERE Prenotazione.AUTO = Auto;
12
13
    commit work;
    ELSE
14
    CALL printf ('[ERRORE] NON ESISTE UN VEICOLO DA TE PRENOTATO CON QUESTA TARGA');
15
16
    END IF;
17
    END
18
19
    DELIMITER;
```

#### 5.3.8 INSERISCI TRAGITTO

```
DELIMITER |
1
    CREATE PROCEDURE inserisciTragitto (IN EmailN varchar (30), IN km smallint,
2
    IN tipe VARCHAR (30), OUT res smallint)
3
    BEGIN
    IF EXISTS ( SELECT EMAILP FROM UTENTE_PREMIUM WHERE EMAILP = EmailN )
    THEN INSERT INTO TRAGITTO (EMAILP, KM, TIPO) values (EmailN, km, tipe);
    SET res = (SELECT LAST_INSERT_ID());
    CALL printf('QUALCOSA PREMIUM è ANDATO STORTO');
    END IF;
10
    IF EXISTS (SELECT EMAILA FROM UTENTE_AZIENDALE WHERE EMAILA = EmailN )
11
    THEN INSERT INTO TRAGITTO (EMAILA, KM, TIPO) values (EmailN, km, tipe);
12
13
    CALL printf ('INSERIMENTO AZINEDALE AVVENUTO');
    SET res = (SELECT LAST_INSERT_ID());
14
15
    CALL printf('QUALCOSA AZIENDALE è ANDATO STORTO');
16
    END IF;
17
    END;
18
19
    DELIMITER;
```

#### 5.3.9 INSERISCI TAPPA

```
DELIMITER |
1
    CREATE PROCEDURE InserisciTappa (IN idN smallint, IN cittaN varchar (20),
2
    IN viaN varchar (30), IN orarioN datetime)
    BEGIN
    DECLARE x SMALLINT;
    SET x = (SELECT CAPIENZA FROM veicolo, tragitto_prenotazione, prenotazione
    WHERE tragitto_prenotazione.ID_TRAG=idN AND
    \verb|tragitto_prenotazione.ID_PREN=prenotazione.ID| AND|
    veicolo.TARGA=prenotazione.AUTO
   limit 1);
10
    IF EXISTS (SELECT ID FROM TRAGITTO WHERE ID = idN)
11
    THEN INSERT INTO TAPPA (ID_TRAGITTO, CITTA, VIA, ORARIO_ARRIVO, POSTI)
12
13
    VALUES (idN, cittaN, viaN, orarioN, X-1);
    ELSE
14
    CALL printf ('[ERRORE], NON ESISTE NESSUN TRAGITTO CON QUESTO ID');
15
16
    END;
17
18
    DELIMITER;
19
```

## 6 CODICE NOSQL

```
function mongoLog($document)
2
3
        if (extension_loaded("mongodb")) {
            try {
                 $mng = new MongoDB\Driver\Manager("mongodb://localhost:27017");
                 $bulk = new MongoDB\Driver\BulkWrite();
                 $document = ['_id' => new MongoDB\BSON\ObjectID(), $document];
                 $bulk->insert($document);
                 $mng->executeBulkWrite('testPHP.LOG', $bulk);
10
            } catch (MongoDB\Driver\Exception\Exception $e) {
11
                 //Errore
12
            }
13
        }
14
15
16
```

Il log deglle operazioni di inserimento di dati nel database è stato gestito tramite delle funzioni in PHP. La difficoltà principale nell'implementare questa funzionalità è stata la configurazione del software, in particolare l'installazione manuale del driver PHP per MongoDB senza utilizzare PEAR o PECL. Esistono infatti due versioni di tale driver: una verisone legacy ben documentata compatibile con PHP 5.x ed una versione più recente compatibile con PHP 7.x ma meno documentata. Ad ogni modo, dopo aver configurato manualmente questa estensione l'implementazione è risultata relativamente semplice; le informazioni che vengono salvate nel log sono la query che è stata eseguita, ottenuta dall'oggetto PDO e il risultato dell'esecuzione (se disponibile). Questi dati vengono inseriti in un array che viene passato ad una funzione che si occupa della connessione, di inserire l'array in un documento valido e di eseguire l'inserimento del log.

Giacomo Minello

Matteo Tramontano

Davide Menetto